UNIVERSIDAD DE VALLADOLID

E. U. de Informática (Segovia)

Grado en Ingeniería Informática de Servicios y Aplicaciones

Gestor de Prácticas de Alumnos UVa

(GdPU)

Alumno: Avelino de Dios Costa

Tutor: Fernando Díaz Gómez
Luis Ignacio Sebastián Martín

BLOQUE I

MEMORIA DEL TFG

ÍNDICE DEL DOCUMENTO

1 INTRODUCCIÓN

1.1 IDENTIFICACIÓN DEL TRABAJO FIN DE GRADO

Título: Gestor de Prácticas de Alumnos UVa (GdPU).

Autor: Avelino de Dios Costa.

Directores: Fernando Díaz Gómez.
Luis Ignacio Sebastián Martín.

Departamento: Informática.

Área: Ciencias de la Computación e Inteligencia Artificial.

1.2 ORGANIZACIÓN DE LA DOCUMENTACIÓN

La documentación del GdPU se va a organizar basándose en la estructura recomendada por la Escuela Universitaria de Informática de Segovia. En concreto, dividiremos la documentación en tres bloques o secciones independientes: la memoria del proyecto (Bloque I), la documentación técnica (Bloque II) y la documentación de usuario (Bloque III). Esta organización se justifica en que cada sección tiene una audiencia bien diferenciada en cada caso (el público en general, programadores, desarrolladores y analistas, y usuarios de la aplicación, respectivamente) y representan un documento autocontenido (con entidad propia), por lo que aparecen como subdocumentos independientes, cada uno con su propio índice, y separados convenientemente dentro del volumen encuadernado.

El Bloque I, denominado "Memoria del TFG" contiene, además de la introducción e identificación del TFG:

- La descripción general del trabajo fin de grado (TFG) en la que se tratan los objetivos perseguidos por el mismo, motivaciones, así como una serie de cuestiones metodológicas y tecnológicas relevantes.
- La descripción general del producto software desarrollado, describiéndose las funcionalidades soportadas, una descripción de la arquitectura adoptada y de su despliegue en el entorno de explotación final.
- La planificación y presupuesto del trabajo en base a la estimación de la carga de trabajo siguiendo el método COCOMO.
- Las cuestiones de diseño reseñables en el desarrollo del trabajo y que condicionaron la elaboración de la documentación técnica.
- Las cuestiones de implementación reseñables en el desarrollo del trabajo con el fin de lograr un producto final de calidad.

- Las conclusiones y posibles ampliaciones del presente trabajo.

- La bibliografía y un anexo en el que se incluye la licencia del producto software desarrollado.

El Bloque II, denominado "Documentación Técnica", aporta los detalles suficientes a desarrolladores, programadores y analistas para una profunda comprensión de cómo se ha diseñado e implementado la aplicación. Esta sección se ha estructurado en los siguientes apartados:

- Análisis del Sistema.

  Se incluye información detallada de los objetivos perseguidos por el sistema, así como los requisitos de información, los casos de uso, actores del sistema y los requisitos no funcionales.

- Diseño del Sistema.

  En el apartado de diseño se ha incluido el diseño de la base de datos (tanto con el modelo entidad/relación, como con el grafo relacional), el diseño del GdPU con la descripción de sus tipos de objetos y asociaciones, y el modelo de comportamiento.

- Pruebas del sistema.

  En este último apartado se incluye tanto el tipo, como la batería de pruebas realizadas.

El Bloque III, denominado "Documentación de usuario", contiene el manual de instalación y uso de la aplicación. Esta sección tiene como fin formar al usuario de la aplicación en su uso de forma sencilla y gráfica.

1.3 Estructura del CD

El CD que acompaña a este documento tiene la siguiente estructura:

```
CD/DVD
├── DOCUMENTACIÓN
│   ├── BLOQUE I - MEMORIA_DEL_TFG
│   │   └── BLOQUE I – Memoria_TFG.pdf
│   ├── BLOQUE II - DOCUMENTACION_TECNICA
│   │   └── BLOQUE II – Manual_Tecnico.pdf
│   ├── BLOQUE III - MANUAL_DE_USUARIO
│   │   └── BLOQUE III – Manual_Usuario.pdf
│   └── OTROS
│       ├── Planifiacion_GdPU.pod
│       ├── UML_GdPU.uml
│       └── Ent_Rel.dia
└── SOFTWARE
    ├── CÓDIGO_FUENTE
    │   ├── GdPU
    │   └── OTROS
    │       └── IMAGENES-ICONOS
    └── EJECUTABLES
        ├── EXTENSIÓN EXE
        │   └── Gestor de Practicas.exe
        ├── EXTENSIÓN JAR
        │   └── Gestor de Practicas.jar
        └── INSTALADOR
            └── GdP-Setup.exe
```

2 DESCRIPCIÓN GENERAL DEL TFG

Hasta ahora, la gestión de la documentación relacionada con la asignación de prácticas externas en empresa por parte de la Escuela de Ingeniería Informática de Segovia se hacía una manera manual, es decir, mediante el archivo de documentos físicos en los que constaba la relación de la empresa y la práctica (convenios marco, definición de ofertas de práctica, etc.).
Si bien, todo este método de gestión sigue funcionando correctamente, el presente proyecto está dedicado a desarrollar una aplicación que proporcione las funcionalidades necesarias para gestionar las prácticas externas en empresa a los alumnos (solicitantes) de la Escuela de Ingeniería de Informática de Segovia. Para ello, la aplicación a desarrollar deberá ser una aplicación de escritorio, con una interfaz sencilla fácilmente manejable y con la posibilidad de ejecutarse en cualquier sistema operativo.

Otra característica fundamental de la aplicación a desarrollar será disponer de una base de datos embebida, es decir, una base de datos incrustada en el propio programa a la que se pueda tener acceso sin necesidad de ningún tipo de conexión a internet.

2.1 OBJETIVOS

El objetivo de este proyecto es facilitar la gestión realizada por la secretaría administrativa de la Escuela de Ingeniería Informática de Segovia en lo que concierne a los alumnos que soliciten o busquen prácticas en empresa. Para ello se desarrollará una aplicación de escritorio la cual ayude a tramitar de una manera eficiente y ordenada dichas solicitudes.

Como consecuencia de ese objetivo, tenemos:

- **Gestión de Alumnos:** El software permitirá la creación/modificación/eliminación de alumnos en función del año académico que curse.
- **Gestión de Empresas:** El software permitirá la creación/modificación/eliminación de empresas con convenio de prácticas con la Universidad y la posibilidad de adjuntar el pdf de dicho convenio.
- **Gestión de Prácticas ofertadas por empresas:** El software permitirá la creación/modificación/eliminación de prácticas ofertadas por las empresas con convenio de prácticas y la posibilidad de adjuntar el pdf de dicha oferta.
- **Gestión de Profesores:** El software permitirá la creación/modificación de profesores que tutelen prácticas en empresa.
- **Asignación de Prácticas*:*** El software permitirá la asignación de una práctica a un alumno seleccionado y la posibilidad de adjuntar múltiples pdf de informes y cualquier otra información relacionada con el desempeño de la práctica.

2.2 CUESTIONES METODOLÓGICAS

El objetivo del desarrollador de este trabajo es cumplir estrictamente con los plazos de entrega, manteniendo el alcance del proyecto establecido en una primera fase de análisis. Para ello, es imprescindible utilizar una metodología de trabajo eficaz, adaptada a las características del proyecto, cuyas claves son:

- **Máxima dedicación.**

  El desarrollador del proyecto aplicará la máxima dedicación posible durante la duración del mismo.

- **Mecanismos de comunicación ágiles.**

  La constante y rápida comunicación entre el tutor del trabajo y el desarrollador garantizará la resolución de los posibles problemas que se presenten durante el desarrollo del proyecto y, en consecuencia, el éxito del proyecto.

Al tratarse de un sistema con módulos de funcionamiento independientes, la **metodología para el desarrollo** de software que se adecua más y por lo tanto se utilizará será el **desarrollo basado en prototipos**. Este permite desarrollar modelos de aplicaciones de software que permiten ver la funcionalidad básica de la misma, sin necesariamente incluir toda la lógica o características del modelo terminado. El desarrollo basado en prototipos permite al cliente evaluar en forma temprana el producto, e interactuar con los diseñadores y desarrolladores para saber si se está cumpliendo con las expectativas y las funcionalidades acordadas.

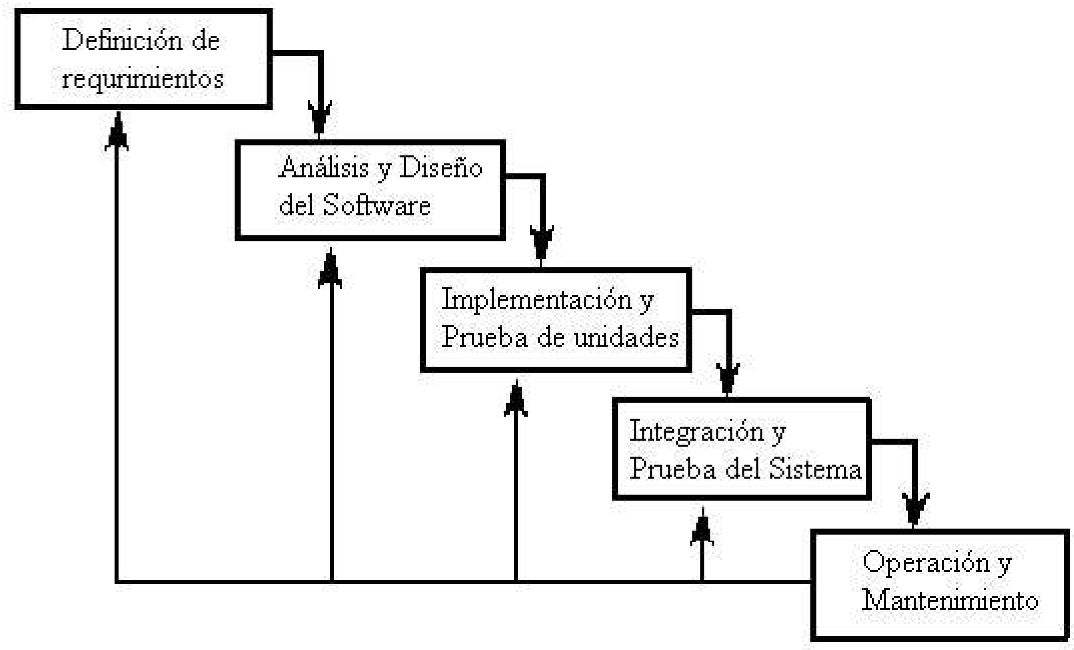
Dentro de los ciclos de vidas iterativos, se eligió el modelo incremental. Los motivos más importantes para realizar esta elección son:

- Estar basado en el ciclo de vida en cascada y en la construcción de prototipos.
- Priorización de requisitos. Los más críticos se incluyen en los primeros incrementos.
- Los primeros incrementos sirven como prototipos para la detección de nuevos requisitos.
- Riesgo bajo de fallar en el proyecto total.

En el modelo de cascada (cada fase empieza cuando se ha terminado la anterior, es decir, se han cumplido cada uno de los objetivos establecidos para esa fase) se definen las siguientes etapas que deben cumplirse de forma sucesiva:

- Especificación de requisitos
- Diseño del software
- Construcción o Implementación del software
- Integración
- Pruebas (o validación)
- Despliegue (o instalación)
- Mantenimiento

Figura 01: Modelo en cascada



Para el modelado de datos (descripción del modelo) se ha utilizado UML (Lenguaje Unificado de Modelado) que es el lenguaje de modelado de sistemas de software más utilizado en la actualidad, respaldado por el OMG (Object Management Group).

En cuanto a los artefactos entregables del análisis y diseño del sistema, presentes en la documentación técnica de este proyecto, se han seguido las plantillas recomendadas en la asignatura de Ingeniería del Software, impartida en nuestra Escuela, intentando ser lo más preciso posible.

Cada uno de los prototipos puede constar de las siguientes etapas (no tienen por qué ser todas):

1. Análisis, que comprende:
   1.1. Identificación de los actores participantes.
   1.2. Identificar requisitos (requisitos de información, restricciones, casos de uso, etc.).
   1.3. Definir detalladamente los requisitos del sistema.

2. Diseño.
    2.1. Diseño de la estructura modular del sistema.
    2.2. Diseño de interfaces entre componentes del sistema.
    2.3. Diseño de la interfaz del usuario.
    2.4. Definición detallada de componentes del sistema.
    2.5. Diseño conceptual y lógico del modelo datos del sistema.

3. Implementación.
    3.1.1. Implementación de los componentes del sistema.

4. Pruebas.
    4.1. Prueba de los componentes desarrollados y del sistema general.
    4.2. Corrección de errores.

2.3 TECNOLOGÍAS DE DESARROLLO

El GdPU ha sido desarrollado casi en su totalidad con tecnologías *OpenSource*. Se ha optado, desde el principio, por este tipo de productos dada su ausencia de licencias, con el consiguiente ahorro económico.

El lenguaje de programación usado para el desarrollo del sistema ha sido el lenguaje de programación orientado a *Java,* lo que hace posible reutilizar sus librerías, a través del IDE Netbeans. Este lenguaje tiene una gran cantidad de librerías que facilitan el desarrollo de aplicaciones, además de estar ampliamente documentado, lo que incluye disponer código de demostración sobre cuestiones básicas del mismo.

Se ha utilizado el lenguaje SQL para la definición, consulta y manipulación de los datos de nuestra base de datos embebida así como la API JDBC (Java Data Base Conectivity) + Apache Derby para el acceso a dicha base de datos desde nuestro software.

Las herramientas usadas para el desarrollo del proyecto son:

- Hardware
    - PC de desarrollo con las características normales de un ordenador actual

- Software
    - “Netbeans 8.01” como entorno de desarrollo.
    - “OpenProj 1.4” para la planificación del proytecto
    - “StarUML 2.1.2” para el modelado de la aplicación usando el lenguaje visual UML
    - “Microsoft Office 2010” para el desarrollo de la documentación y manual de usuario
    - “DIA 0.97.2-2” para la realización del modelo entidad-relación
    - “Adobe Photoshop CS6 Demo” para la creación del icono de la aplicación y documentación.

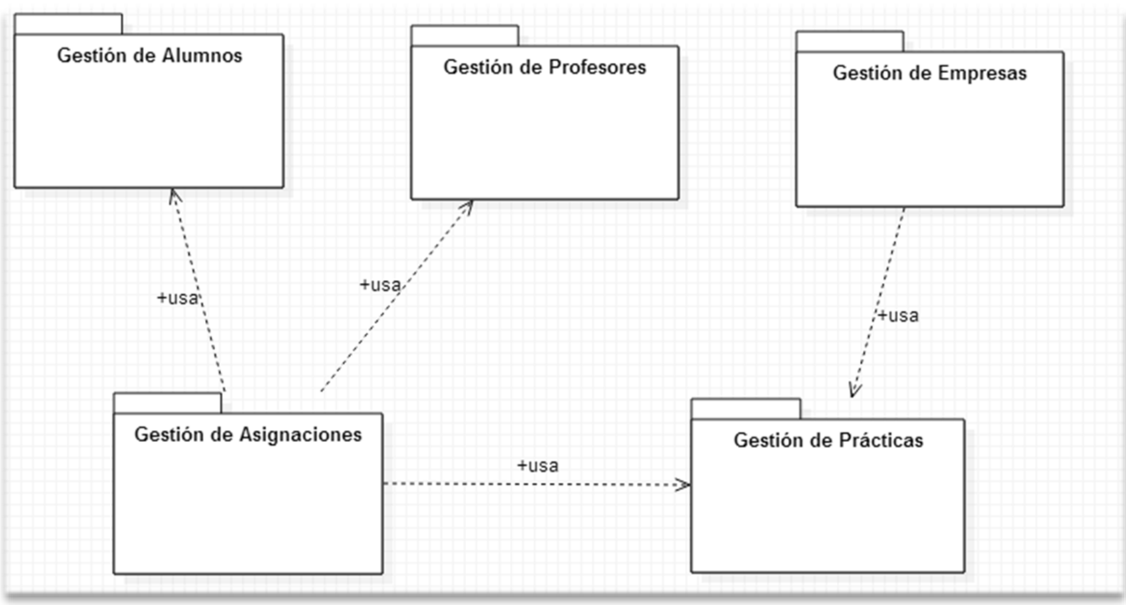
3 DESCRIPCIÓN GENERAL DEL PRODUCTO

La interfaz de usuario, como mediador entre el usuario y la lógica de la aplicación, es de vital importancia a la hora de diseñar y elaborar un nuevo producto software. Aunque a veces olvidada, esta capa es la responsable de la mayoría de éxitos y fracasos de los productos software que salen al mercado. Por ello, para el desarrollo de este producto se ha prestado especial atención en que la interfaz sea eficaz, intuitiva, esté bien estructurada y facilite el trabajo al usuario.

Además de prestar atención al diseño de la interfaz, se ha realizado el desarrollo del producto pensando en el cumplimiento de las siguientes características esenciales:

- Accesibilidad.
- Usabilidad
- Escalabilidad.
- Reusabilidad de código.
- Facilidad de actualización

Figura 02: Módulos del GdPU



Cada uno de los paquetes de "Gestión" contiene la funcionalidad necesaria para para permitir al usuario añadir, consultar, modificar, o borrar los datos de las respectivas secciones de gestión a la que desee acceder.
Todos ellos, además, usaran los paquetes JSWing (JDialog, JFrame, etc) proporcionados por Java para crear la interfaz.

Como podemos observar en la *figura 02,* el paquete de "Gestión de Asignaciones" utiliza, para su funcionamiento, los paquetes de Gestión de Alumnos, Gestión de Profesores y Gestión de Prácticas. Este último, a su vez, necesita del paquete "Gestión de Empresas" para que le proporcione la empresa que desea crear/ofertar la nueva práctica.

3.1 Funcionalidades del producto

Más detalladamente, a continuación, se enumeran las funcionalidades de las que dispone el producto desarrollado:

La funcionalidad de la aplicación resultante del desarrollo especificado en este documento se divide en varias áreas:

- **Gestor de apartados (alumnos, empresas, prácticas y profesores)**

  Los diferentes gestores (denominados como "Gestión de Profesores", etc.) permiten la administración de los datos correspondientes a los diferentes apartados. En concreto, desde el gestor se puede la creación, modificación o eliminación de un registro, así como la posibilidad de consultarlos y listarlos. La persistencia de los datos de acceso se realiza mediante el uso de una base de datos SQL.

- **Asignador de prácticas**

  El software GdPU dispone de un apartado específico, intuitivo y sencillo en el que poder asignar una práctica determinada, a un alumno determinado, bajo la tutela de un profesor seleccionado.

- **Explorador de documentos**

  El software GdPU dispone de unos botones que facilitan la adjunción y visualización de documentos relacionados con el desempeño de la práctica (informes práctica, convenios marco, etc.).

- **Impresor de Tablas**

  El software GdPU dispone de un botón para la impresión de cualquier tabla que desee el usuario.

- **Filtrador de Búsquedas**

  El software GdPU dispone de diferentes formas de filtrar la información mostrada por las tablas de los diferentes gestores con la finalidad de poder facilitar el acceso a una información determinada al usuario.

- **Backup de Datos**

  El software GdPU dispone tanto de un botón para realizar backups de datos (que incluirá la información registrada en la base de datos y sus documentos adjuntos) como de un botón para la restauración del mismo.

- **Interfaz Gráfica**

El software GdPU desarrollado cuenta con una interfaz clara y atractiva que incluye entre otras cosas:

- Menú con iconos que diferencian diferentes secciones dentro de una misma pantalla.
- Diálogos en forma de alertas para comunicar sucesos y errores.
- Interfaz optimizada para cualquier tipo de resolución de pantalla

3.2 Arquitectura del producto

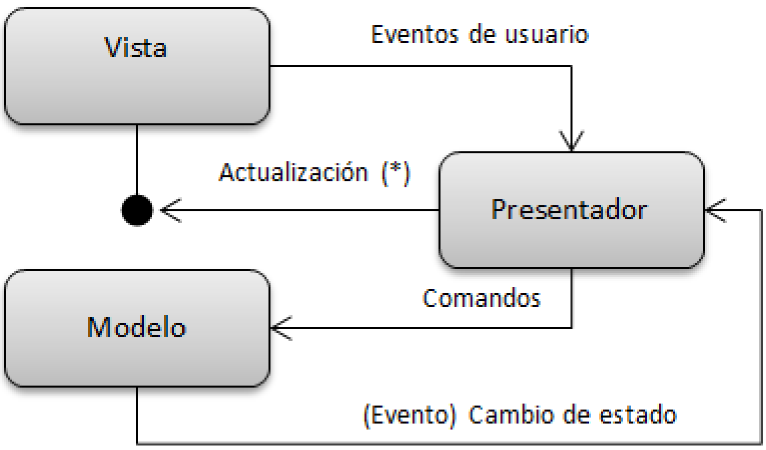
En este apartado se detalla tanto la arquitectura lógica como la arquitectura física del producto desarrollado, denominado abreviadamente GdPU.

3.2.1 Arquitectura lógica

Java, es una plataforma de programación que permite la ejecución de aplicaciones empresariales en arquitecturas de diversos niveles y distribuidas, pero en nuestro caso, al ser una aplicación de escritorio, distinguiremos solo una capa "grande". La capa de presentación que incluirá a su vez la capa de negocio.

Para la capa de presentación hemos utilizado el patrón MVP (Modelo-Vista-Presentador) ya que las interfaces de usuario van a ser de tipo Java, y este tipo de modelo está más orientado hacia este tipo de aplicaciones. El patrón MVP nos permite hacer una clara distinción entre los componentes tiene como objetivo separar la interfaz de usuario de la lógica de aplicaciones, facilitando así la interacción con el usuario. En nuestra aplicación, tenemos por un lado el Presentador y la Vista (prácticamente unidas) en el Paquete "Vistas" y por otro lado, el Modelo en el paquete "Controlador".

Figura 03: Arquitectura Lógica: Modelo-Vista-Presentador



La vista, compuesta por ventanas y controles que forman la interfaz de usuario se encarga de presentar la información de una manera vistosa al usuario.
El presentador. Escucha los eventos que se producen en la vista y ejecuta las acciones necesarias a través del modelo. Además puede tener acceso a las vistas a través de las interfaces que la vista debe implementar.
El modelo es donde se lleva a cabo toda la lógica de negocio y operaciones. Incorpora la capa de dominio y persistencia, encargada de guardar los datos en un medio persistente (BD, registro, etc.), en nuestro caso, la conexión a la base de datos es proporcionada por JDBC (Java Database Connection) utilizando la extensión Derby.

El concepto de este patrón es bastante sencillo. Por un lado la vista, se encarga de mostrar la información al usuario y de interactuar con él para hacer ciertas operaciones. Por otro lado, el modelo, ignorante de cómo la información es mostrada al usuario, realiza toda la lógica de las aplicaciones usando las entidades del dominio. Y por último tenemos al presentador que es el que “presenta” a ambos actores sin que haya ningún tipo de dependencia entre ellos.

El flujo normal de nuestra aplicación, usando el patrón MVP como se puede ver en la Figura anterior, es fundamentalmente el siguiente:

1. El usuario interactúa con la interfaz de usuario de alguna manera (por ejemplo rellenando un formulario y pulsando el botón “Registrar").

2. El presentador toma los datos introducidos mediante los cuadros de texto y valida que el contenido de estos es el correcto (en el caso de no ser correcto, el presentador se lo comunica al usuario a través de la vista)

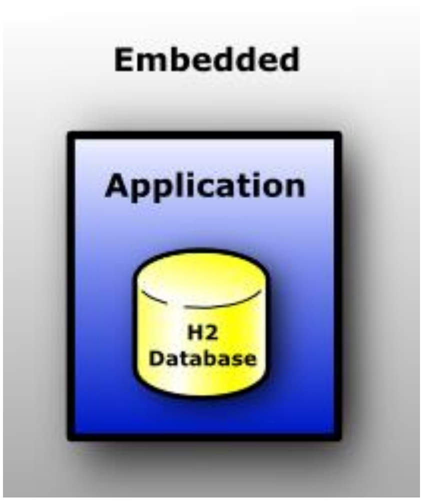
3. Si es correcto, el presentador realizaría las operaciones pertinentes usando el modelo.

4. Al recibir el resultado de la operación, el presentador mostraría el resultado a través de la vista. En este caso, mediante un pop-up que dijera “Registro añadido correctamente”.

3.2.2 ARQUITECTURA FÍSICA

Lo primero que debemos hacer antes de definir e implantar la arquitectura física es saber que el número de capas (layers) de la arquitectura lógica tiene que ser aproximadamente igual al número de capas (tiers) de la arquitectura física.
Pero, al ser nuestra aplicación un tipo de aplicación de escritorio con la base de datos embebida y sin acceso a internet, podemos afirmar que solo poseerá 1 capa física, sin necesidad de utilizar ningún firewall, proporcionando así una aplicación que destaca por su portabilidad, uno de los requisitos fundamentales que se buscaba al inicio de este proyecto.

Figura 04: Arquitectura Física: Aplicación con BBDD embebida



4 PLANIFICACIÓN Y PRESUPUESTO

4.1 ESTIMACIÓN DE TRABAJOS

Como punto de partida para la estimación de los trabajos del proyecto, es necesario calcular el tamaño que previsiblemente tendrá nuestro sistema.

Existen varias técnicas de estimación que se basan en las características que debe tener el sistema a desarrollar, pero para este proyecto se va a utilizar la estimación por puntos de función.

4.1.1 ESTIMACIÓN MEDIANTE PUNTOS DE FUNCIÓN (PFA)

La estimación por puntos de función se realiza para obtener una aproximación del tamaño que pueda tener el proyecto. Esta estimación será usada más adelante para evaluar el esfuerzo y coste que supondrá desarrollar el proyecto.

4.1.1.1 EXPLICACIÓN DEL PROCESO

Antes de realizar la estimación se va a describir qué es y cómo se va a realizar.

El método de estimación se basa en una métrica que cuantifica la funcionalidad que se debe entregar al usuario al construir la aplicación [Albrech, 1979]. Para ello, este método utiliza unos parámetros que sirven para evaluar dicha funcionalidad. Estos parámetros son:

- Número de entradas: Datos que el usuario aporta al sistema (nombre de ficheros, menús de selección, etc.).
- Número de salidas: Datos que el sistema aporta al usuario (informes, mensajes).
- Número de ficheros lógicos internos: Ficheros o bases de datos internos al sistema (es decir sólo los utiliza el sistema, ficheros maestros).
- Número de ficheros externos: Ficheros o bases de datos externos al sistema, es decir, que pueden ser "vistos" por otras aplicaciones.
- Número de consultas externas: Entradas que requieren de una respuesta por parte del sistema.

Los pasos a seguir son:

- ❖ Se debe contar el número de elementos de cada clase.
- ❖ Cada elemento debe ser clasificado según su grado de complejidad (alta, media o baja).
- ❖ Por último se obtienen los PFNA (Puntos de Función No Ajustados) mediante una suma ponderada de esas cantidades con los pesos que aparecen a continuación:

Tabla 01: Multiplicadores de complejidad de la estimación por puntos de función

Parámetro significativo	Complejidad baja	Complejidad media	Complejidad alta
Entradas	x3	x4	x6
Salidas	x4	x5	x7
Ficheros internos	x7	x10	x15
Ficheros externos	x5	x7	x10
Consultas externas	x3	x4	x6

Los criterios para evaluar la complejidad de los elementos de cálculo son los siguientes:

Tabla 02: Tabla para valorar la complejidad de las características del proyecto.

		Datos elementales		
Ficheros lógicos externos e internos.	**Registros elementales**	**1-9**	**20-50**	**>51**
	1	Baja	Baja	Media
	2-5	Baja	Media	Alta
	>6	Media	Alta	Alta
		Datos elementales		
Salidas y consultas	**Tipos de ficheros**	**1-5**	**6-19**	**>20**
	0-1	Baja	Baja	Media
	2-3	Baja	Media	Alta
	>4	Media	Alta	Alta
		Datos elementales		
Entradas	**Tipos de ficheros**	**1-4**	**5-15**	**>16**
	0-1	Baja	Baja	Media
	2-3	Baja	Media	Alta
	>3	Media	Alta	Alta

Una vez calculados los parámetros y su complejidad los sumamos mediante la siguiente suma ponderada:

PFNA = (Nº Entradas × multiplicador (complejidad))
+ (Nº Salidas × multiplicador (complejidad))
+
(Nº Fichero internos × multiplicador (complejidad)) +
(Nº Ficheros externos × multiplicador (complejidad))
+ (Nº Consultas externas × multiplicador
(complejidad)).

Los PFNA deben ser ajustados mediante un Factor de Ajuste (FA).

El factor de ajuste se obtiene de la suma de los 14 factores de complejidad (FC), que se describen más adelante, mediante la siguiente ecuación:

$$FA = (0{,}01 \times \Sigma\ FC) + 0{,}65$$

Los puntos de función ajustados (PF) se obtienen de la siguiente forma:

$$PF = PFNA \times FA$$

Existen 14 factores que contribuyen a la complejidad de una aplicación, cada uno de

ellos valorados dentro de una escala de 0 a 5.

FACTORES DE COMPLEJIDAD[6]:

1. Comunicación de Datos

0 Aplicación es *batch* exclusivamente.
1-2 Impresión o entrada de datos remota.
3-5 Teleproceso (TP) interactivo.
3 TP interface a un proceso *batch*.
5 La aplicación se interactiva.

2. Rendimiento (referido a la importancia de respuesta dentro de todo el sistema)

0-3 Análisis y diseño de las consideraciones del rendimiento son estándar. No se requieren requerimientos especiales por parte del usuario.
4 En la fase de diseño se incluyen tareas del análisis del rendimiento para cumplir los requerimientos del usuario.
5 Además se utilizan herramientas de análisis del rendimiento en el diseño, desarrollo e instalación.

3. Frecuencia de Transacciones

0-3 Las tasas son tales que las consideraciones de análisis de rendimiento son estándares.
4 En la fase de diseño se incluyen tareas de análisis de rendimiento para verificar las altas tasas de transacciones.
5 Además se utilizan herramientas de análisis del rendimiento.

4. Requisitos de manejo del usuario final.

0 Sistema *batch*.
1-3 No se especifican requerimientos especiales.
4 Se incluyen tareas de diseño para la consideración de factores humanos.
5 Además se utilizan herramientas especiales o de prototipado para promover la eficiencia.

5. Procesos complejos

¿Qué características tiene la aplicación?
Mucho procesamiento matemático y/o lógico
Muchas excepciones de procesamiento, muchas transacciones incompletas y mucho reprocesamiento de las transacciones
Procesamiento de seguridad y/o control sensitivo

0 No se aplica nada de esto.
1-3 Se aplica alguna cosa.
4 Se aplican dos cosas.
5 Se aplica todo.

6. Facilidad de mantenimiento e instalación

0-1 No se requieren por parte del usuario facilidades especiales de conversión e instalación.

2-3 Los requerimientos de conversión e instalación fueron descritos por el usuario y se proporcionaron guías de conversión e instalación.

4-5 Además se proporcionaron y probaron herramientas de conversión e instalación.

7. Instalación en múltiples lugares. Añadir puntos por cada uno de los siguientes factores:

0 El usuario no requiere la consideración de más de un puesto

1 De uno a cuatro puestos.

2 Cinco o más puestos.

1 Se proporciona documentación y plan de apoyo para soportar la aplicación en varios lugares.

2 Los puestos están en países diferentes.

8. Funciones distribuidas. "Distribuida" significa que los componentes de la aplicación están distribuidos en dos o más procesadores diferentes.

0 La aplicación no ayuda a la trasferencia de datos o a la función de procesamiento entre los componentes del sistema.

1 La aplicación prepara datos para el usuario final de otro procesador.

2-3 Los datos se preparan para trasferencia, se trasfieren y se procesan en otro componente del sistema.

4 Igual que 2-3, pero con realimentación al sistema inicial.

5 Las funciones de procesamiento se realizan dinámicamente en el componente más apropiado del sistema.

9. Gran carga de trabajo (referente a la importancia del entorno)

0-3 La aplicación corre en una máquina estándar sin restricciones de operación.

4 Restricciones de operación requieren características específicas de la aplicación en el procesador central.

5 Además hay restricciones específicas a la aplicación en los componentes distribuidos del sistema.

10. Entrada interactiva de datos

0-2 Hasta el 15% de las transacciones tienen entrada interactiva. 3-4 15% al 30% tienen entrada interactiva.

5 30% al 50% tienen entrada interactiva.

11. Actualizaciones On-Line

1 Nada.

1-2 Actualización *on-line* de los ficheros de control. El volumen de actualización es bajo y la recuperación fácil.

3 Actualización *on-line* de la mayoría de los ficheros internos lógicos.

4 Además es esencial la protección contra la pérdida de datos.

5 Además se considera el coste de recuperación de volúmenes elevados.

12. Utilización con otros sistemas (el código se diseña para que sea compartido o utilizable por otras aplicaciones. No confundir con 13).

0-1 Una aplicación local que responde a las necesidades de una organización usuaria.

2-3 La aplicación utiliza o produce módulos comunes que consideran más necesidades que las del usuario.

4-5 Además, la aplicación se "empaquetó" y documentó con el propósito de fácil reutilización.

13. Facilidad de Operación

0 No se especifican por parte del usuario consideraciones específicas de operación.

1-2 Se requieren, proporcionan y prueban procesos específicos de arranque, *backup* y recuperación.

3-4 Además la aplicación minimiza la necesidad de actividades manuales, tales como instalación de cintas y papel.

5 La aplicación se diseña para operación sin atención.

14. Facilidad de Cambio (esfuerzo específico de diseño para facilitar cambios futuros). Añadir puntos por cada uno de los siguientes factores:

0-2 No hay requerimientos especiales del usuario para minimizar o facilitar el cambio.

3-4 Se proporciona capacidad de consulta flexible.

5 Datos importantes de control se mantienen en tablas que son actualizadas por el usuario a través de procesos on-line interactivos.

Por último, podemos estimar las líneas de código en función de los puntos de función. Como cada lenguaje tiene sus particularidades, las líneas de código equivalentes a un punto de función no pueden ser las mimas para todos los lenguajes. Por ello Casper Jones realizó una tabla con las equivalencias que vemos a continuación.

Tabla 03: Tabla de correspondencia entre puntos de función y Líneas De Código (LDC).

Lenguaje	*LDC/PF*
Ensamblador	320
C	150
Cobol	106
Pascal	91
Basic	64
TCL	64
Java	**53**
C++	29

4.1.1.2 Estimación por puntos de función

Comenzamos obteniendo los parámetros que nos permitirán evaluar la funcionalidad del desarrollo software.

Entradas (9):

- Complejidad baja (9):
  - Selección de gestor.
  - Datos de alumno.
  - Datos de empresa.
  - Datos de prácticas.
  - Datos de profesores.
  - Selección del archivo convenio empresa.
  - Selección del archivo convenio prácticas.
  - Selección de archivo informes prácticas.
  - Selección de tabla a imprimir.

Salidas (3):

- Complejidad baja (3):
  - Mensaje de errores.
  - Tablas de datos.
  - Tabla impresa.

Ficheros internos (1):

- Complejidad baja (1):
  - Base de datos.

Ficheros externos (1):

- Complejidad media(1)
  - Manual de instalación y de usuario.

Consultas externas (4):

- Complejidad baja (2):
  - Consulta de tablas.
  - Consulta con filtro de tablas.

- Complejidad media (2):
  - Explorar directorio local.
  - Visualización de archivo seleccionado.

Para obtener los PFNA se realiza la suma de los productos del número de parámetros de cada tipo por su multiplicador de complejidad.

$$PFNA = (9 \times 3) + (3 \times 4) + (1 \times 7) + (1 \times 7) + (2 \times 3) + (2 \times 4) = 67$$

Obtenemos el Factor de ajuste valorando los factores de complejidad.

Tabla 04: Tabla con la valoración de los factores de complejidad.

Factores de complejidad (FC)	0-5	Factores de complejidad (FC)	0-5
Comunicación de datos	5	Funciones distribuidas	0
Rendimiento	1	Gran carga de trabajo	3
Frecuencia de transacciones	1	Entrada on-line de datos	0
Requisitos de manejo de usuario final	4	Actualización on-line	0
Procesos complejos	1	Utilización con otros sistemas.	4
Facilidad de mantenimiento	3	Facilidad de operación	1
Instalación en múltiples lugares	0	Facilidad de cambio	2

$$\Sigma\ FC = 24$$
$$FA = (0{,}01 \times 25) + 0{,}65 = 0{,}90$$

Los puntos de función ajustados (PF) se obtienen de la siguiente forma:

$$PF = PFNA \times FA = 67 \times 0{,}90 = 60{,}3$$

Ahora calculamos las líneas de código *Java* por cada punto de función. Para ello, sacamos la equivalencia de la tabla que realizó Casper Jones.

1PF = 53 Líneas de código en *Java*

LDC = 60,3 x 53 ≈ 3196 Líneas de código *Java*

4.1.2 ESTIMACIÓN DE COSTES POR COCOMO (CONSTRUCTIVE COST MODEL)

En este apartado se realiza una estimación del esfuerzo y tiempo que supondrá realizar el proyecto software. Para ello, este método se basa en una estimación previa del tamaño del software en líneas de código (LDC).

4.1.2.1 EXPLICACIÓN DEL ALGORITMO COCOMO

El algoritmo de COCOMO varía en función de las características del sistema que se va a desarrollar. En concreto, este modelo de estimación diferencia entre sistemas orgánicos, empotrados y semi-libres.

La Tabla 05 muestra los tipos de desarrollos y sus correspondientes valores para los parámetros A, B y C de las expresiones utilizadas para estimar el esfuerzo y tiempo de desarrollo.

Tabla 05: Modos de desarrollo contemplados por COCOMO.

Modo de desarrollo	A	B	C
Orgánico	3,20	1,05	0,38
Empotrado	3,00	1,12	0,35
Semi-Libre	2,80	1,20	0,32

Fórmulas:

- Esfuerzo nominal [personas • mes] = $A \times (KLDC)^{B}$
- Esfuerzo [personas • mes] = Esfuerzo Nominal × Π Factores de coste
- Tiempo de desarrollo [meses] = $2{,}5 \times Esfuerzo^{C}$
- Nº medio de personas [personas] = Esfuerzo / Tiempo de desarrollo

Los factores de coste para el cálculo del esfuerzo se obtienen de la siguiente tabla:

Tabla 6: Tabla con los valores correspondientes a los factores de coste que intervienen en el cálculo del esfuerzo.

Factores	**Valor de los factores**					
	Muy bajo	Bajo	Medio	Alto	Muy alto	Extra
Fiabilidad requerida	0,75	0,88	1	1,15	1,4	
Tamaño de la base de datos		0,94	1	1,08	1,16	
Complejidad del software	0,70	0,85	1	1,15	1,30	1,65
Restricciones de tiempo de ejecución			1	1,11	1,30	1,66
Restricciones de memoria			1	1,06	1,21	1,56
Volatilidad del hardware		0,87	1	1,15	1,30	
Restricciones de tiempo de respuesta		0.87	1	1,07	1,15	
Calidad de los analistas	1,46	1,19	1	0,86	0,71	
Experiencia con el tipo de aplicación	1,29	1,13	1	0,91	0,82	
Experiencia con el hardware	1,21	1,10	1	0,90		
Experiencia con el lenguaje de programación.	1,14	1,07	1	0,95		
Calidad de los programadores	1,42	1,17	1	0,86	0,70	
Técnicas modernas de programación	1,24	1,10	1	0,91	0,82	
Empleo de herramientas	1,24	1,10	1	0,91	0,83	
Restricciones a la duración del proyecto	1,23	1,08	1	1,04	1,10	

4.1.2.2 APLICACIÓN DE COCOMO

Para este proyecto se elige el modo orgánico puesto que este modo esta aconsejado en:

- Desarrollos con entorno estable.
- No demasiada innovación técnica.
- Escasas presiones de tiempo.
- Tamaño relativamente pequeño (<50 KLDC, Kilo Líneas De Código).

La estimación de KLDC para este proyecto es de 3,196 KLDC. (Estimación obtenida por puntos de función).

Esfuerzo nominal = $3{,}2 \times (3{,}196)^{1{,}05}$ = 10,84 personas • mes

Esfuerzo = 10,84 × 1,15 (Fiabilidad requerida) × 0,85 (Complejidad software) × 1,06 (Restricciones de memoria) × 0,86 (calidad de los analistas) × 0,91 (experiencia con la aplicación) × 1,10 (Experiencia con el hardware) × 0.95 (experiencia con el lenguaje de programación) × 1,07 (Restricciones de tiempo de respuesta) × 0,86 (calidad de los programadores) × 0,83 (Empleo de herramientas) = 7,01 personas • mes.

Tiempo de desarrollo = $2.5 \times 7{,}01^{0{,}38}$ = 5,24 meses

Mediante el esfuerzo y el tiempo de desarrollo se puede hace una estimación del número de personas necesarias cada mes para el desarrollo.

Nº de personas = 7,01 personas • mes / 5,24 meses = 1.34 personas al mes para realizar el proyecto en 5.24 meses.

Dado que el proyecto únicamente va a ser realizado por una persona, se va a extender el tiempo de desarrollo de los 5.24 meses a 6 meses.

4.2 PLANIFICACIÓN

En este apartado vamos a detallar la planificación temporal del presente proyecto.

Desde la planificación temporal, el trabajo es dividido en una serie de unidades que podemos medir de forma cuantitativa. Estas unidades son las tareas. Cada tarea tiene una duración determinada y su inicio y fin pueden estar condicionados por el resto de tareas del desarrollo. De esta forma, podemos definir que una tarea no pueda ser iniciada hasta que se haya completado otra, o bien, que dos tareas puedan ser iniciadas a la vez.

Para el proyecto que se describe en este documento, las tareas identificadas y planificadas se pueden observar en *la Figura 04*.

Figura 05: Tareas planificadas y calendarizadas

		Nombre	Duración	Inicio	Terminado
1		⊟Gestor de Prácticas UVa	140 days?	10/12/14 8:00	23/06/15 17:00
2		⊟ITERACIÓN 1	102 days	10/12/14 8:00	30/04/15 17:00
3		⊟PLANIFICACIÓN	12 days	10/12/14 8:00	25/12/14 17:00
4	✔	Estudio del proyecto a desarrollar	2 days	10/12/14 8:00	11/12/14 17:00
5		Búsqueda y recopilación de información	4 days	12/12/14 8:00	17/12/14 17:00
6		Estudio de Metodologías	4 days	18/12/14 8:00	23/12/14 17:00
7		Estimación de la duración del proyecto	2 days	24/12/14 8:00	25/12/14 17:00
8		⊟ANÁLISIS	7 days	26/12/14 8:00	5/01/15 17:00
9		⊟Elaboración del DRS	7 days	26/12/14 8:00	5/01/15 17:00
10		Especificación de requisitos y restricciones	3 days	26/12/14 8:00	30/12/14 17:00
11		Desarrollo de Casos de Uso	2 days	31/12/14 8:00	1/01/15 17:00
12		Especificación de Casos de Uso	2 days	2/01/15 8:00	5/01/15 17:00
13		⊟DISEÑO	4 days	6/01/15 8:00	9/01/15 17:00
14		⊟Elaboración del DAS	4 days	6/01/15 8:00	9/01/15 17:00
15		Diseño del diagrama de clases	1 day	6/01/15 8:00	6/01/15 17:00
16		Elabroación de diagramas	2 days	7/01/15 8:00	8/01/15 17:00
17		Diseño del Diagrama Entidad-Relación	1 day	9/01/15 8:00	9/01/15 17:00
18		⊟IMPLEMENTACIÓN	69 days	12/01/15 8:00	16/04/15 17:00
19		Estudio de Lenguajes y Tecnologías	4 days	12/01/15 8:00	15/01/15 17:00
20		Diseño gráfico de la interfaz	15 days	16/01/15 8:00	5/02/15 17:00
21		Creación del modelo de datos	2 days	6/02/15 8:00	9/02/15 17:00
22		⊟Desarrollo de Aplicación	48 days	10/02/15 8:00	16/04/15 17:00
23		Creación de gestor de alumnos	7 days	10/02/15 8:00	18/02/15 17:00
24		Creación de gestor de empresas	7 days	19/02/15 8:00	27/02/15 17:00
25		Creación de gestor de prácticas	7 days	2/03/15 8:00	10/03/15 17:00
26		Creación de gestor de profesores	7 days	11/03/15 8:00	19/03/15 17:00
27		Creación de gestor de asignaciones	7 days	20/03/15 8:00	30/03/15 17:00
28		Creación Listar Tablas	5 days	31/03/15 8:00	6/04/15 17:00
29		Creación y recuperación de Backpus	8 days	7/04/15 8:00	16/04/15 17:00
30		PRUEBAS	10 days	17/04/15 8:00	30/04/15 17:00
31		⊟ITERACIÓN 2	37 days	1/05/15 8:00	22/06/15 17:00
32		⊟ANÁLISIS	1 day	1/05/15 8:00	1/05/15 17:00
33		Ampliación y revisión del documento DRS	1 day	1/05/15 8:00	1/05/15 17:00
34		⊟DISEÑO	1 day	4/05/15 8:00	4/05/15 17:00
35		Ampliación y revisión del documento DAS	1 day	4/05/15 8:00	4/05/15 17:00
36		⊟IMPLEMENTACIÓN	19 days	5/05/15 8:00	29/05/15 17:00
37		Aumento de funcionalidades y operaciones de	7 days	5/05/15 8:00	13/05/15 17:00
38		Revisión y Mejora de código	3 days	14/05/15 8:00	18/05/15 17:00
39		Revisión y mejora de interfaz Gráfica	9 days	19/05/15 8:00	29/05/15 17:00
40		PRUEBAS	2 days	1/06/15 8:00	2/06/15 17:00
41		⊟DOCUMENTACIÓN	12 days	3/06/15 8:00	18/06/15 17:00
42		Realización de Manual de Instalación	2 days	3/06/15 8:00	4/06/15 17:00
43		Realización de Manual de Usuario	2 days	5/06/15 8:00	8/06/15 17:00
44		Realización de Memoria del Proyecto	8 days	9/06/15 8:00	18/06/15 17:00
45		⊟FORMACIÓN	2 days	19/06/15 8:00	22/06/15 17:00
46		Formación de Usuarios	2 days	19/06/15 8:00	22/06/15 17:00

Como se puede observar, el proyecto se ha dividido en dos grandes iteraciones:

1 **Iteración 1:**

Esta es la iteración principal, engloba 102 días de desarrollo y trascurre desde la planificación hasta las pruebas iniciales. La fase mas larga de esta iteración, es la implementación y más particularmente todo lo que engloba el desarrollo de la aplicación (como es de esperar).

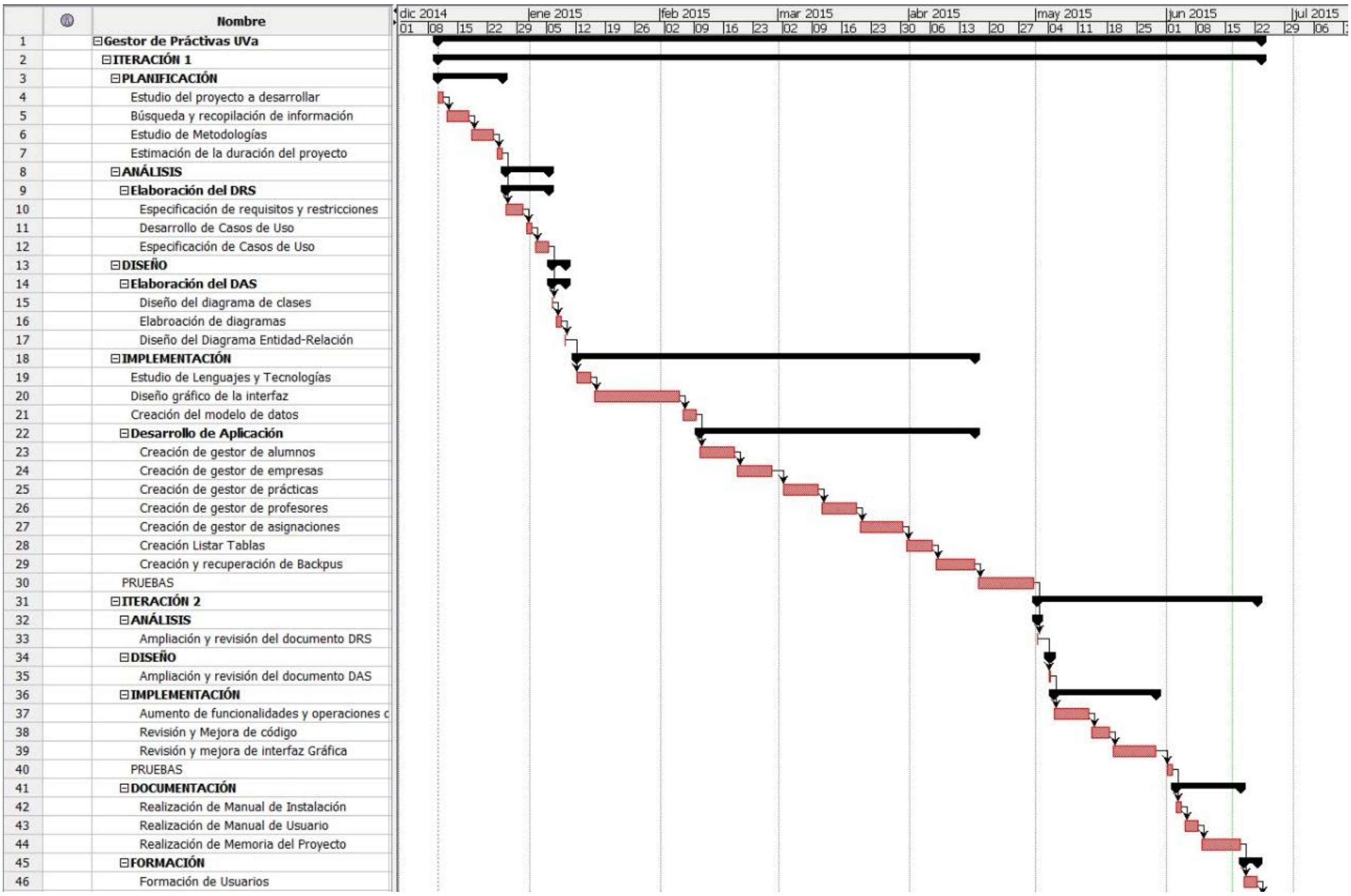
2 Iteración 2:

Esta iteración está destinada sobre todo a la ampliación de documentación y corrección de errores, que nos hayamos ido encontrando a lo largo de la primera iteración.
Como novedad respecto a la anterior, esta iteración tiene la fase de documentación que, si bien esta se ha ido desarrollando a lo largo de todo el proyecto, hasta los momentos finales del mismo no se ha puesto en conjunto en un mismo documento, y se refiere al desarrollo de la presente documentación + el manual de usuario, y la fase de formación, que consiste en explicar y/o formar al usuario que se encargará de utilizar este software.

La duración del proyecto es de 140 días. El comienzo fue el 10 de Diciembre de 2014 y la fecha en la que se finaliza el desarrollo es el 22 de Junio de 2015.

La *Figura 06* muestra el diagrama de Gantt, donde se observan las dependencias entre las tareas.

Figura 06: Diagrama de GANTT con la dependencia de tareas



4.3 PRESUPUESTO

4.3.1 PRESUPUESTO ESTIMADO

Anteriormente, mediante el método de puntos de función y COCOMO se había estimado que el tiempo para la realización de este proyecto sería de 6 meses.

Por lo tanto:

-Recursos Humanos:

Partimos de la base de que el sueldo de un Ingeniero Informático aproximadamente es de 20.000 €/año.
Como el año tiene 12 meses, el sueldo mensual sería 20.000/12 = 1833,33 €

Como nuestra estimación nos decía que requeriríamos de 6 meses de trabajo, 1833,33€ x 6 = **11.000 €**

-Recursos Hardware

Ordenador para el desarrollo, implantación y pruebas de nuestra aplicación
Conexión a internet para desarrollo de la aplicación, obtención de información y descarga de software requerido.
Impresora para imprimir la documentación y manual de nuestra aplicación.

Tabla 06: Presupuesto Estimado: Recursos Hardware.

HARDWARE	USO (%)	COSTE TOTAL (€)	UNIDADES	COSTE (€)
Ordenador personal	*25,00%*	*1.300,00 €*	*28€ x 1*	*28,00 €*
Conexión a internet	*100,00%*	*20 € / mes*	*1*	*20,00 €*
Impresora	*4,00%*	*80,00 €*	*1*	*4,00 €*
TOTAL:		***76,00 €***		

Suponemos que un ordenador cuesta 1.300€ y que tiene una vida útil de 4 años, como el proyecto no dura 1 año completo si no que dura X meses, aplicando una regla de tres, el coste del ordenador al proyecto supone 28 €/mes por unidad.

28 euros aproximadamente el equipo al mes. Como son 6 meses de uso: 168 €

Respecto a la conexión a internet, suponemos que es de 20€ / mes, como el proyecto X meses, la conexión a internet supone un coste de, 6 € x 20 €/mes = 120 € aproximadamente.

La impresora tiene un valor inicial de 80€ y un ciclo de vida de 4 años (si su uso es elevado), como el proyecto tiene una duración de meses, y si se suman los ocasionados (tinta, papel, reparaciones), el coste se incrementa a 10 € aproximadamente.
En resumen: Ordenador personal + acceso Internet + Impresora (en % uso) =168+120+10= **298€**

-Recursos Software

Tabla 07: Presupuesto Estimado: Recursos Software

SOFTWARE	USO (%)	COSTE TOTAL (€)	COSTE POR UNIDAD (€)	COSTE (€)
Windows 7	*25,00%*	*120,00 €*	*30,00 €*	*30,00 €*
Microsoft Office 2010	*10,00%*	*130,00€*	*13,00€*	*13,00€*
StarUML	-	*Software Libre*	*0,00€*	*0,00€*
Google Chrome	-	*Software Libre*	*0,00€*	*0,00€*
Adobe Reader 8	-	*Software Libre*	*0,00€*	*0,00€*
Netbeans 8 IDE	-	*Software Libre*	*0,00€*	*0,00€*
OpenProject 1.4	-	*Software Libre*	*0,00€*	*0,00€*

***TOTAL:* 43,00€**

Uso de Windows 7 del 25% + Microsoft Office al 5% + Multitud programas Opensource = 43 €

Tabla 08: Presupuesto Estimado: Coste total

	COSTE
Hardware	298
Software	43
Desarrollo	11.000
TOTAL	11.341 €

4.3.2 PRESUPUESTO TOTAL

Hemos visto mediante el diagrama de Gantt que el tiempo de desarrollo de la aplicación han sido 140 días. 140 días son aproximadamente 4,67 meses

Por lo tanto:

-Recursos Humanos:

Partimos de la base de que el sueldo de un Ingeniero Informático aproximadamente es de 20.000 €/año.
Como el año tiene 12 meses, el sueldo mensual sería 20.000/12 = 1833,33 €

Como nuestra estimación nos decía que requeriríamos de 4,67 meses de trabajo, 1833,33€ x 6 = **8555,54 €**

-Recursos Hardware

Ordenador para el desarrollo, implantación y pruebas de nuestra aplicación
Conexión a internet para desarrollo de la aplicación, obtención de información y descarga de software requerido.
Impresora para imprimir la documentación y manual de nuestra aplicación.

Tabla 09: Presupuesto Total: Recursos Hardware

HARDWARE	USO (%)	COSTE TOTAL (€)	UNIDADES	COSTE (€)
Ordenador personal	*25,00%*	*1.300,00 €*	*28€ x 1*	*28,00 €*
Conexión a internet	*100,00%*	*20 € / mes*	*1*	*20,00 €*
Impresora	*4,00%*	*80,00 €*	*1*	*4,00 €*
TOTAL:		***76,00 €***		

Suponemos que un ordenador cuesta 1.300€ y que tiene una vida útil de 4 años, como el proyecto no dura 1 año completo si no que dura X meses, aplicando una regla de tres, el coste del ordenador al proyecto supone 28 €/mes por unidad.

28 euros aproximadamente el equipo al mes. Como son 4,67 meses de uso: 130,67 €

Respecto a la conexión a internet, suponemos que es de 20€ / mes, como el proyecto X meses, la conexión a internet supone un coste de, 4,67 € x 20 €/mes = 93,4 € aproximadamente.

La impresora tiene un valor inicial de 80€ y un ciclo de vida de 4 años (si su uso es elevado), como el proyecto tiene una duración de meses, y si se suman los ocasionados (tinta, papel, reparaciones), el coste se incrementa a 10 € aproximadamente.
En resumen: Ordenador personal + acceso Internet + Impresora (en % uso) =168+120+10= **234,07€**

-Recursos Software

Tabla 10: Presupuesto Total: Recursos Software

SOFTWARE	USO (%)	COSTE TOTAL (€)	COSTE POR UNIDAD (€)	COSTE (€)
Windows 7	*25,00%*	*120,00 €*	*30,00 €*	*30,00 €*
Microsoft Office 2010	*10,00%*	*130,00€*	*13,00€*	*13,00€*
StarUML	-	*Software Libre*	*0,00€*	*0,00€*
Google Chrome	-	*Software Libre*	*0,00€*	*0,00€*
Adobe Reader 8	-	*Software Libre*	*0,00€*	*0,00€*
Netbeans 8 IDE	-	*Software Libre*	*0,00€*	*0,00€*
OpenProject 1.4	-	*Software Libre*	*0,00€*	*0,00€*

***TOTAL:* 43,00€**

Uso de Windows 7 del 25% + Microsoft Office al 5% + Multitud programas Opensource = 43 €

Tabla 11: Presupuesto Total: Coste Total

	COSTE
Hardware	243,
Software	43
Desarrollo	8555,54

TOTAL 8.832,61 €

4.3.3 Desviación de presupuesto

Por último, debemos comprobar la desviación entre el presupuesto estimado y el presupuesto real, para ello no tenemos nada más que hacer:

11.341-8.832,61 = **2508,39 €**

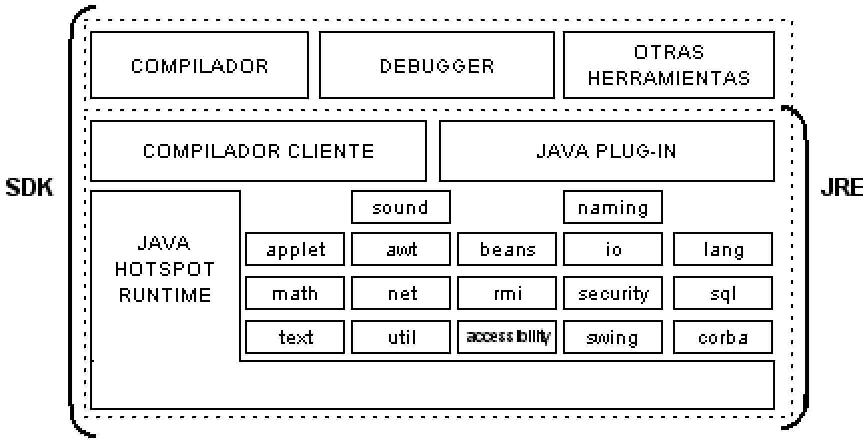
Como podemos observar, nos hemos ahorrado unos 2500 € respecto a lo que habíamos estimado, que hubieran correspondido al sueldo del ingeniero durante el periodo de 1,3 meses de más que habíamos estimado respecto a lo real.

5 Cuestiones de diseño reseñables

➢ Plataforma utilizada

La plataforma de software utilizada para nuestro Gestor de Prácticas de Alumnos es JavaSE, anteriormente conocido como Java Plataform 2, Standard Edition o J2SE (edición estándar de Java).
J2SE es la edición principal de la plataforma Java y provee de las capacidades de desarrollo y ejecución de software escrito en el Lenguaje Java.
J2SE incluye herramientas y APIs (como son como son *java.applet, java.beans, java.awt, java.rmi, java.security, java.sql (usado en este proyecto) o javax.swing*) para desarrollar aplicaciones con interfaz gráfica, acceso a base de datos, acceso a directorios, seguridad, entrada/salida, programación en red y varias otras funcionalidades.

Figura 07: Plataforma utilizada: J2SE



Las aplicaciones de J2SE se ejecutan en una máquina virtual de Java (Java Runtime Environment) ofreciendo funcionalidad común y proporcionando dos características muy importantes para una aplicación: Portabilidad y Escalabilidad.

J2SE nos proporciona algunas ventajas como escalabilidad (aunque varíe la magnitud del proyecto, no disminuirá el rendimiento), fiabilidad, madurez y seguridad (desde 1997 con multitud de proyectos a sus espaldas), alta productividad, y ser distribuida (soporta múltiples SO, ya que se ejecuta donde se pueda ejecutar una Máquina Virtual Java).

➢ API UTILIZADA

Java Database Connectivity (JDBC) es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

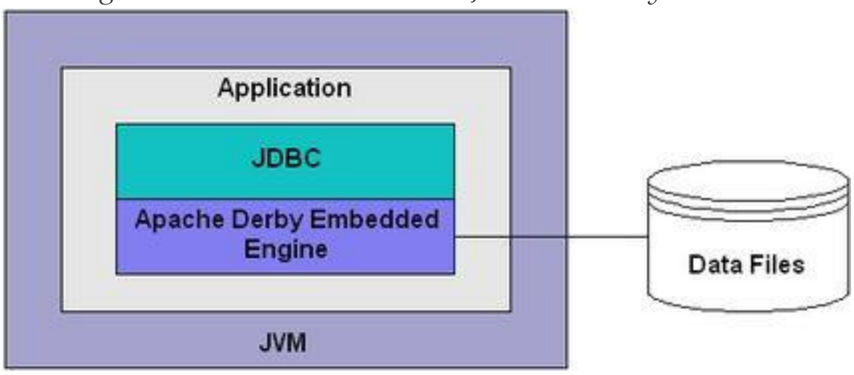
El API JDBC se presenta como una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos. Un manejador de conexiones hacia un modelo de base de datos en particular es un conjunto de clases que implementan las interfaces Java y que utilizan los métodos de registro para declarar los tipos de localizadores a base de datos (URL) que pueden manejar. Para utilizar una base de datos particular, el usuario ejecuta su programa junto con la biblioteca de conexión apropiada al modelo de su base de datos, y accede a ella estableciendo una conexión; para ello provee el localizador a la base de datos y los parámetros de conexión específicos. A partir de allí puede realizar cualquier tipo de tarea con la base de datos a la que tenga permiso: consulta, actualización, creación, modificación y borrado de tablas, ejecución de procedimientos almacenados en la base de datos, etc.

➢ EXTENSIÓN A JDBC: DERBY

Apache Derby es un sistema gestor de base de datos relacional escrito en Java que puede ser empotrado en aplicaciones Java y utilizado para procesos de transacciones online. Tiene un tamaño de 2 MB de espacio en disco. Inicialmente distribuido como IBM Cloudscape, Apache Derby es un proyecto open source licenciado bajo la Apache 2.0 License. Actualmente se distribuye como Sun Java DB.

Derby es una base de datos relacional implementada íntegramente en Java bajo licencia Apache. Tiene un tamaño muy reducido y viene incluida en el JDK de Java desde la versión 6. Esto permite realizar aplicaciones, y distribuirlas simplemente como un archivo .jar, lo que es bastante útil, sobre todo a la hora de realizar pruebas de concepto.
A continuación vemos una imagen de la arquitectura de Derby, que prácticamente nos servirá como base en el desarrollo de nuestra aplicación.

Figura 08: Funcionamiento de JDBC + Derby



MODELO DE DATOS

El modelo de datos del GdPU es bastante simple, únicamente es necesario almacenar la información de los diferentes módulos de gestión. Por ello, se ha diseñado la aplicación para usar una base de datos *SQL* en la que se almacenan todos estos datos.

Los motivos para usar una base de datos *SQL* en vez de un simple archivo son fundamentalmente dos:

- El tratamiento de los datos en SQL es transparente para el desarrollador, mientras que en el caso de usar un archivo sería necesario diseñar e implementar un formato de almacenamiento de los datos.

- La facilidad de escalar una base de datos SQL, cuando sea necesario por ampliación de funcionalidad, es mucho mayor que cuando se trata de un archivo. Además, la posibilidad de que se efectúen ampliaciones en el gestor de prácticas es muy alta.

En definitiva, el uso de un sistema u otro en este desarrollo no supone a día de hoy una gran ventaja, ya que no se va a guardar una gran cantidad de información, ni van a existir relaciones complejas entre los datos. Pero en un futuro, sí pueden existir más datos que almacenar y más relaciones entre ellos y, con el diseño actual, se tendría una buena base para continuar el desarrollo.

A pesar de la simplicidad de la base de datos, el diseño de clases para acceder a la información no sigue el mismo principio, de hecho, el diseño realizado permitiría acceder a cualquier base de datos *SQL* implementando unos sencillos métodos que retornen un código SQL que realice las operaciones necesarias. Con la posibilidad, incluso de que esta BD dejara de estar embebida y se situara en un servidor remoto.

6 Conclusiones y posibles ampliaciones

6.1 Conclusiones

Tras realizar el presente proyecto, las conclusiones positivas que he sacado son las siguientes:

Facilidad de desarrollo

Una de las principales ventajas del desarrollo de aplicaciones *Java* es que no es necesario crear un nuevo lenguaje de programación desde cero, sino que permite al desarrollador reutilizar librerías creadas para otros proyectos. Además, como veremos a continuación, existe mucha documentación para realizar casi cualquier tarea y para todo lo demás, existe una comunidad de desarrolladores que da soporte a cualquier persona.

Sumado a todo esto, tenemos un buen entorno de desarrollo como es Netbeans que se integra completamente con el kit de desarrollo J2SE, y que permite realizar el autocompletado de código, que facilita mucho la tarea cuando no se conoce en profundidad el lenguaje de programación.

Facilidad de adaptación

Otra de las ventajas que proporciona *Java* en su desarrollo de aplicaciones es que cualquier aplicación desarrollada en este lenguaje puede ser ejecutada en cualquier sistema operativo que tenga una JVM (Java Virtual Machine), por lo que, por medio del instalador, aseguramos que eta se instale.

Mucha documentación y excelente comunidad de desarrolladores

La documentación oficial de *Java* que está disponible para los desarrolladores es muy abundante y de buena calidad. El desarrollador de *Java* dispone de mucha información, desde aspectos básicos de desarrollo, hasta detalles muy concretos de la plataforma. Además, casi todo está documentado con ejemplos, con lo que la adaptación a la plataforma es muy rápida por parte del desarrollador.

Adicionalmente, por si toda la información oficial de *Java* no fuera suficiente, existe una gran comunidad de desarrolladores que, de forma totalmente altruista, dan soporte a los problemas que se puedan presentar a cualquier desarrollador, ya sea principiante o experto.

Como aspectos negativos destaca:

POCA INFORMACIÓN RESPECTO A BASE DE DATOS EMBEBIDAS

Si bien Java tiene una gran comunidad de desarrolladores, y librerías reutilizables a sus espaldas, el "mundillo" de aplicaciones con base de datos embebidas es cada día que pasa menor, por lo que es difícil encontrar información al respecto.
Esto es debido a que cada vez se usa más internet (prácticamente cualquier aparato electrónico/informático dispone de este tipo de conexión) y se prefiere tener una base de datos guardada en un servidor remoto, obteniendo así mayor seguridad ante cualquier desperfecto que pueda sufrir tu aparato físico. Esta ventaja, a la vez es un inconveniente puesto que sin acceso a internet, no podríamos acceder a estos datos lo que implicaría no poder usar la aplicación.
No obstante, se ha conseguido salvar este problema y se ha conseguido desarrollar una aplicación diferente, que en eso consistía la realización de este proyecto.

6.1.1 ADQUISICIÓN Y APLICACIÓN DE CONOCIMIENTOS

Para poder realizar el proyecto, resultó indispensable aplicar los conocimientos que se adquirieron durante el trascurso de mi formación académica, sobre todo en lo referente a Java.

Además, el desarrollo de la aplicación, como ya se ha comentado anteriormente, se ha realizado sobre la plataforma *Java*. Antes de comenzar este proyecto, mi conocimiento sobre *Java* estaba limitado a los conocimientos proporcionados por las asignaturas de "Programación Orientada a Objetos" y "Programación y Estructura de Datos". Por este motivo, tuve la necesidad de investigar y aprender sobre el uso de interfaces para la generación de una aplicación de escritorio interactiva, donde se aprendieron aspectos básicos del uso de estas interfaces y que me estimularon para adentrarme en la plataforma. Gracias a ello pude familiarizarme más aún una plataforma muy potente que sigue estando presente en la actualidad y que tiene un futuro muy prometedor.

Durante el desarrollo del proyecto, la utilización de conocimientos adquiridos en las distintas asignaturas de la carrera se fue sucediendo. Entre los conocimientos destacados están el lenguaje de modelado UML, el diseño y creación de bases de datos, la base de la programación orientada a objetos, la gestión de un proyecto, etc.

6.2 POSIBLES AMPLIACIONES

Algunas de las posibles ampliaciones y mejoras, que a mi juicio se podrían llevar a cabo en un futuro próximo, se enumeran a continuación:

- **BASE DE DATOS EN SERVIDOR REMOTO**

  Con unas pequeñas modificaciones de código, se podría lograr que la base de datos estuviera en un servidor remoto y no embebida (o embebida solo para copias de seguridad)

- **AMPLIACIÓN A TODOS LOS DEPARTAMENTOS DE LA SECRETARÍA DE LA UNIVERSIDAD**

Esta aplicación dejaría de ser solo para la Escuela de Ingeniería Informática de Segovia y añadiendo pequeños cambios (como una base de datos global, que englobe el resto de base de datos por departamento) podría ser usado por toda la secretaría del centro.

- **USO CONCURRENTE**

Como consecuencia de los dos anteriores apartados, se podría hacer un acceso concurrente a los datos de nuestra base de datos en el servidor, para que múltiples usuarios ejecutaran sus acciones al mismo tiempo.

- **FICHERO REGISTRO (HISTÓRICO DE DATOS)**

Y como consecuencia de lo anterior, si muchas personas utilizaran la aplicación al mismo tiempo, se requeriría de un .txt que registrara cada una de las acciones realizadas en el sistema con la finalidad de llevar un control sobre todas las operaciones.

7 BIBLIOGRAFÍA

The Apache DB Project. *QuickStar* [en línea]. Actualizada: 15 de Abril de 2014. [Fecha de consulta: Diciembre 2014]. Disponible en: http://db.apache.org/derby/quick_start.htmlx

Oracle Technology Network. *Get Started* [en línea].
[Fecha de consulta: Diciembre-Junio 2015]. Disponible en: http://www.oracle.com/technetwork/topics/newtojava/documentation/index.html

Oracle Technology Network. ***The Java™ Tutorials***. *Usos de Swing* [en línea]. [Fecha de consulta: Noviembre-Junio 2015]. http://docs.oracle.com/javase/tutorial/uiswing/

Programación.net. *Portal de programación en castellano.* Acceso a Base de Datos JDBC [en línea]. Actualizada: 28 de Junio de 2011. [Fecha de consulta: Noviembre-Junio 2015]. Disponible en:
http://programacion.net/articulo/acceso_a_bases_de_datos_[jdbc]_93/8
http://docs.oracle.com/javase/tutorial/uiswing

Rumbaugh, J., Jacobson, I., Booch, G. (1999). *El lenguaje unificado de modelado. Manual de referencia*. Madrid: Pearson Educación, S.A., 2000.

Apuntes y material de Proceso de Desarrollo de Software [Francisco José Cabrera]
Disponible en: http://campusvirtual.uva.es/

Apuntes y material de Plataformas de Software Empresarial [Anibal Bregón Bregón]
Disponible en: http://campusvirtual.uva.es/

Apuntes y material de Progammazione II [Proff. Marcugini]
Disponible en: http://www.informatica.unipg.it/

8 ANEXOS

8.1 ÍNDICE DE FIGURAS

8.2 ÍNDICE DE TABLAS

8.3 GLOSARIO DE TÉRMINOS

BBDD O BD

Abreviatura bases de datos. Se define una base de datos como una colección estructura de datos entre los que existen relaciones lógicas y ha sido diseñada para satisfacer los requerimientos de información de una empresa u organización.

BBDD EMBEBIDA

Una Base de Datos Embebida o Empotrada es aquella que no inicia un servicio en nuestra máquina independiente de la aplicación, pudiéndose enlazar directamente a nuestro código fuente o bien utilizarse en forma de librería.”
“Una base de datos embebida es un archivo que colocas en la carpeta de tu aplicación y que corre dentro de tu proceso, siendo únicamente accesible por éste.

UML

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group).

Es un lenguaje Figura para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

IDE

Un ambiente de desarrollo interactivo o entorno de desarrollo integrado (en inglés Integrated development environment) es una aplicación de software, que proporciona servicios integrales para facilitarle al programador de computadora el desarrollo de software. Proporciona un editor de código fuente, herramientas de construcción automáticas y un depurador.

JSWING

El paquete Swing es parte de la JFC (Java Foundation Classes) en la plataforma Java. La JFC provee facilidades para ayudar a la gente a construir GUIs. Swing abarca componentes como botones, tablas, marcos, etc...

Las componentes Swing se identifican porque pertenecen al paquete javax.swing.

GUIs

La interfaz gráfica de usuario, conocida también como GUI (inglés GraphicalUuser Interface) es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos Figuras para representar la información y acciones disponibles en la interfaz. Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una máquina o computador.

API

La interfaz de programación de aplicaciones (IPA), abreviada como API (del inglés: Application Programming Interface), es el conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas.

JVM

Una máquina virtual Java (en inglés Java Virtual Machine, JVM) es una máquina virtual de proceso nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el bytecode Java), el cual es generado por el compilador del lenguaje Java.

SQL O STRUCTURED QUERY LANGUAGE (LENGUAJE ETRUCTURADO DE CONSULTA)

Se trata de un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en éstas con el fin de recuperar, de una forma sencilla, información de interés de una base de datos, así como también hacer cambios sobre ella. Es un lenguaje de cuarta generación (4G

9 LICENCIA DEL PROYECTO

Se ha decidido que el proyecto se distribuya bajo licencia GNU GPL (General Public License) versión 3. Se puede consultar la licencia completa bajo estas líneas o desde http://www.gnu.org/licenses/gpl-3.0.html.

GNU GENERAL PUBLIC LICENSE V3

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps:

(1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed

erroneously to authors of previous versions.
Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or

receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms

your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a

relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section

7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a

charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source.

Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the

Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work

material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License.

You are not responsible for enforcing compliance by third parties with this LicensAn "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe

are valid.
If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIOS